Addressing Large Models:

Multiple and Hierarchical Universality

Meir Feder
School of Electrical Engineering
Tel-Aviv University

Two Messages

- **An information theoretic framework to machine learning:**

  Under the assumption of a given "Hypothesis Class"
  - Replace the "PAC" Criterion
  - Suggests information theoretic learnability measures
  - Based on universal source coding, universal prediction

- **A proposal (with a conjecture) for the "over-parameterized" case where the hypothesis class is not given, or it is ``too large''**

  Multiple and hierarchical universality
  - Learn both a model and a class (or hierarchy of classes)
  - Learnability with non uniform convergence
  - Examples of success from universal source coding. For other learning problems
  - The essence of modern learning: Why DNN, transformers and similar can "explain", "generalize. Open questions..

Universality

Model Independent Schemes

Yet, Strive to Attain Optimal, Model dependent, Performance

An over 30 years Journey

- Many works in the 90's with **Neri Merhav**: universal coding and prediction.

  Later, universal channel decoding

- **Amos Lapidoth** on universal decoding

- **Nadav Shulman**'s less known work on rateless codes for universal channel and joint source/channel coding

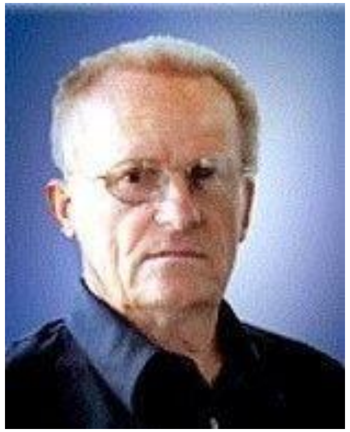
- **Yuval Lomnitz** on universal channel coding with feedback, individual channels

- Many students during the years: **Ofer Shayevitz, Nir Weinberger, Eado Meron, Amir Ingber, Elona Erez, Ronen Dar**, **Zvi Reznic, Nir Elkayam**, others

- Recent works on universal learning: **Yaniv Fogel, Koby Bibas, Shachar Shayovitz, Uriya Pesso, Adi Hendel**

- Re-examining aspects of universal coding/prediction with **Yury Polyianskiy, Amichai Painsky**

Inspiration and Mentors



Jorma Rissanen

Tom Cover

Jacob Ziv

Lossless Source Coding

Online Prediction with Log-Loss

The source coding problem

- Encode a source symbol x or more generally a source sequence
$$\underline{x} = x^n = x_1, \dots, x_n$$

- Existent and known source probability $P(\underline{x})$:

  - With Huffman or arithmetic coding:
    encode each source sequence by $-\log P(\underline{x})$ bits (within 1-2 bits)

  - Expected codelength per source symbol (within $O\left(\frac{1}{n}\right)$ bits): **The entropy**

What can be done when P is unknown or even nonexistent?

Universal source coding

- Suppose a "coding probability" $Q(\underline{x})$.
    - Any source coder may correspond to a probability assignment with

$$Q(\underline{x}) = 2^{-L(\underline{x})}, \quad L(\underline{x}) \text{ the codelength*}$$

- With true P the expected codelength: $H(P) + D(P||Q)$
- How to choose Q optimally?

  When P is unknown or even non-existent

* satisfying Kraft's inequality with equality

The concept of Universal Probability

A single, universal $Q(\underline{x})$

Can be used no matter what $P(\underline{x})$ is, even if it is non-existent!

1. Universality w.r.t a model class

2. Universality w.r.t a very large class of models (all ergodic sources..)

3. **Twice/Multiple/Hierarchical universality**

The equivalence of coding and on-line prediction with log-loss

- Data Compression => Prediction:
  - A coding probability (possibly universal) $Q(\underline{x}) = 2^{-L(\underline{x})}$, $L(\underline{x})$ the codelength
  - Can use this probability assignment for prediction by the chain rule:
    $$Q(\underline{x}) = \prod_{t=1}^{n} Q(x_t|x^{t-1})$$
  - To predict the symbol x_t given the past observation x^{t-1} simply use:
    $$b_t = Q(x_t = x\,|x^{t-1}) = \frac{Q(x^{t-1}x)}{Q(x^{t-1})} = \frac{\sum_{x_{t+1}^n} Q(x^{t-1}x x_{t+1}^n)}{\sum_{x_t^n} Q(x^{t-1}x_t^n)}$$
  - The accumulated log-loss over the sequence $\underline{x}$ is the codelength, $-\log Q(\underline{x})$

The equivalence of coding and on-line prediction with log-loss

- Prediction => Data compression:
  - $x_1 \, x_2 \ldots x_n$ is the data to encode, from a finite alphabet A
  - The (deterministic) action b_t is a probability vector assigned to x_t
    $b_t = \{q_t(\, . \mid x_1 \, x_2 \ldots x_{t-1})\}$
  - The loss: $\ell(b_t, x_t) = -\log q_t(x_t \mid x_1 \, x_2 \ldots x_{t-1})$ is the ideal codelength for encoding x_t .

    Given the assigned distribution, an arithmetic coder can generate a code word with ideal code length $\ell(b_t, x_t)$
  - The accumulated loss is the total code length. It is also $-\log$ of the probability assigned to the entire sequence $x_1 \, x_2 \ldots x_n$, i.e,

$$-\log Q(x^n) = -\log \prod_{t=1}^{n} q_t(x_t|x^{t-1}) = -\sum_{t=1}^{n} \log q_t(x_t|x^{t-1})$$

GENERATIVE AI MODELS LIKE GPT…

ARE LEARNED BY SUCH ON-LINE PREDICTION

Universal Prediction with General Loss

- By coding (or prediction with log-loss) generate $q(x_{t+1}|x^t)$
- Apply "optimal decision" using the universal predicted probability?

$$\hat{b}_{t+1} = \arg\min_{b} \mathbb{E}_{q\left(x_{t+1}\middle|x^t\right)} \ell(b, x)$$

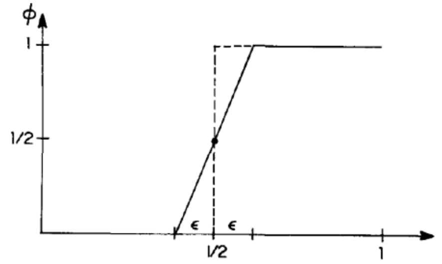
Universal Prediction with General Loss

- By coding (or prediction with log-loss) generate $q(x_{t+1}|x^t)$

- Apply "optimal decision" using the universal predicted probability?

$$\hat{b}_{t+1} = \arg\min_b \mathbb{E}_{q\left(x_{t+1}\middle|x^t\right)} \ell(b, x)$$

- **Not always!**
  For example, for 0-1 loss need to "randomize" decision
  (F-Merhav-Gutman 92)



- **General Solution**: Follow the "Perturbed Probability" (F-Lomnitz, 2013)

Universality with Respect to a Given Model Class

Classical Universal Coding: w.r.t a Model Class

A set of models, $\{P_\theta(x^n)\}, \theta \in \Theta$. “Hypotheses Class”.

- Stochastic setting:
    - x^n is generated by some model $P_\theta \in \Theta$.
- Stochastic mis-specified setting (sometimes “PAC setting”):
    - x^n is generated by some model P, not necessarily in Θ.
- Individual setting:
    - x^n is an arbitrary individual sequence.

Classical Universal Coding: w.r.t a Model Class

A set of models, $\{P_\theta(x^n)\}, \theta \in \Theta$. "Hypotheses Class".

- Stochastic setting:
  - x^n is generated by some model $P_\theta \in \Theta$.

- Stochastic mis-specified setting (sometimes "PAC setting"):
  - x^n is generated by some model P, not necessarily in Θ.

- Individual setting:
  - x^n is an arbitrary individual sequence.

The Elephant in the room: How to choose the model class?

Universal Coding w.r.t a Model Class Θ

- Criteria:
  - Stochastic setting:
    - Look for a universal assignment Q that minimizes the worst case “redundancy”

$$\boxed{min_Q max_\theta\ \mathbb{E}_{P_\theta} \log\frac{P_\theta}{Q} = min_Q max_\theta D(P_\theta||Q)}$$

  - Stochastic mis-specified setting:
    - Even if P is known, cannot avoid: $min_{\theta\in\Theta} D(P||P_\theta) \ = D(P||P_{\theta(P)})$

$$D(P||Q) = min_\theta D(P||P_\theta) + \sum P \log\frac{P_\theta}{Q}$$

    - Look for:

$$\boxed{min_Q max_P \mathbb{E}_P \log\frac{P_{\theta(\mathrm{P})}}{Q}}$$

  - Individual setting:
    - If x^n is known, can attain: $min_{\theta\in\Theta}[-\log P_\theta(x^n)] = -\log P_{\theta^*(x^n)}(x^n)$
    - Look for:

$$\boxed{min_Q max_{x^n} \log\frac{P_{\theta*}}{Q}}$$

The stochastic setting solution

- A Bayesian mixture, with a prior $w(\theta)$ over Θ:

$$Q(x^n) = \int w(\theta)\, P_\theta(x^n) d\theta$$

- The Redundancy-Capacity Theorem (Gallager, Davisson, others, mid 70's) :

$$\min_Q \max_\theta E_\theta \log \frac{P_\theta(x^n)}{Q(x^n)} = \min_Q \max_\theta D(P_\theta || Q) =$$

$$= \max_{w(\theta)} I(\Theta; X^n) = C(\Theta \to X^n) = C_n(\Theta)$$

The stochastic setting solution

- A Bayesian mixture, with a prior $w(\theta)$ over Θ:

$$Q(x^n) = \int w(\theta)\, P_\theta(x^n) d\theta$$

- The Redundancy-Capacity Theorem (Gallager, Davisson, others, mid 70's) :

$$\min_Q \max_\theta E_\theta \log \frac{P_\theta(x^n)}{Q(x^n)} = \min_Q \max_\theta D(P_\theta || Q) =$$

$$= \max_{w(\theta)} I(\Theta; X^n) = C(\Theta \rightarrow X^n) = C_n(\Theta)$$

Strong version: Rissanen, F-Merhav
C_n is essentially the minimal regret for almost all Θ

The solution in the mis-specified setting

- Assume the true $P = P_\phi$ belongs to a large class Φ. Naturally $\Theta \subseteq \Phi$
- The universal probability is a Bayesian mixture over the large class (F-Ployianskiy 2021, Painsky-F 2021):

$$Q(x^n) = \int_{\phi\in\Phi} w(\phi)\, P_\phi(x^n) d\phi$$

where $w(\phi)$ is

$$\arg max_{w(\phi)} \left[I(\Phi; X^n) - \int_{\phi\in\Phi} w(\phi)\, D(P_\phi || P_{\theta(\phi)}) d\phi \right]$$

Intuitively, $w(\phi)$ concentrates on the class Θ!

Denote the "relative redundancy" (Takeuchi-Barron '98): $F_n(\Theta, \Phi) = min_q max_P \mathbb{E}_P \log \frac{P_{\theta(\phi)}}{Q}$

Clearly:

$$C_n(\Theta) \leq F_n(\Theta, \Phi)$$

The solution in the mis-specified setting

- Assume the true $P = P_\phi$ belongs to a large class Φ. Naturally $\Theta \subseteq \Phi$
- The universal probability is a Bayesian mixture over the large class (F-Polyianskiy 2021, Painsky-F 2021):

$$Q(x^n) = \int_{\phi \in \Phi} w(\phi)\, P_\phi(x^n) d\phi$$

where $w(\phi)$ is

$$\arg max_{w(\phi)} \left[I(\Phi; X^n) - \int_{\phi \in \Phi} w(\phi)\, D(P_\phi || P_{\theta(\phi)}) d\phi \right]$$

Intuitively, $w(\phi)$ *concentrates on the class* Θ*!*

Denote the "relative redundancy" (Takeuchi-Barron '98): $F_n(\Theta, \Phi) = min_q max_P \mathbb{E}_P \log \frac{P_{\theta(\phi)}}{Q}$

Clearly:

$$C_n(\Theta) \le F_n(\Theta, \Phi)$$

A strong version (strong lower bound) can be stated: Only for a small fraction of $\Theta, B(\Theta)$, the fraction of $P's$ projected on $\theta \in B(\Theta)$ with relative redundancy smaller than F_n can be large

The solution in the individual setting

- The *Normalized Maximum Likelihood* (NML) solution (Shtarkov '87):

$$Q(x^n) = \max_\theta P_\theta(x^n) / \int \max_\theta P_\theta(x^n)\, \mathrm{d}x^n$$

Worst case regret (individual redundancy):

$$\Gamma_n(\Theta) = \log \int \max_\theta P_\theta(x^n)\, \mathrm{d}x^n \geq F_n(\Theta, \Phi) \geq C_n(\Theta)$$

The solution in the individual setting

- The *Normalized Maximum Likelihood* (NML) solution (Shtarkov '87):

$$Q(x^n) = \max_\theta P_\theta(x^n) / \int \max_\theta P_\theta(x^n)\, \mathrm{d}x^n$$

Worst case regret (individual redundancy):

$$\Gamma_n(\Theta) = \log \int \max_\theta P_\theta(x^n)\, \mathrm{d}x^n \geq F_n(\Theta, \Phi) \geq C_n(\Theta)$$

A strong version Weinberger-Merhav-F can be stated:
Only for a small fraction of $\Theta, B(\Theta)$, the fraction of sequences whose ML is $\theta \in B(\Theta)$ and have an individual redundancy smaller than Γ_n can be large

Important Observations/Results

- **The universal probabilities on all settings depend on the block size – known horizon**

- For "nice" parametric classes with k parameters, asymptotically

$$C_n(\Theta) = \frac{k}{2}\log\frac{n}{2\pi e} + \log\int_{\Theta} |I(\theta)|^{1/2} d\theta + o(1)$$

$$\Gamma_n(\Theta) = \frac{k}{2}\log\frac{n}{2\pi} + \log\int_{\Theta} |I(\theta)|^{1/2} d\theta + o(1)$$

$$F_n(\Theta, \Phi) \approx C_n$$

Where $I(\theta)$ is the Fisher information matrix

- $\Gamma_n(\Theta)$ is greater than $C_n(\Theta)$ since the reference is different!

Important Observations/Results

- **The universal probabilities on all settings depend on the block size – known horizon**

- For “nice” parametric classes with k parameters, asymptotically

$$C_n(\Theta) = \frac{k}{2}\log\frac{n}{2\pi e} + \log\int_\Theta |I(\theta)|^{1/2}d\theta + o(1)$$

$$\Gamma_n(\Theta) = \frac{k}{2}\log\frac{n}{2\pi} + \log\int_\Theta |I(\theta)|^{1/2}d\theta + o(1)$$

$$F_n(\Theta, \Phi) \approx C_n$$

Where $I(\theta)$ is the Fisher information matrix

- $\Gamma_n(\Theta)$ is greater than $C_n(\Theta)$ since the reference is different!

For “nice” parametric models, *a Bayesian mixture , over Θ with Jeffreys’ prior, proportional to* $|I(\theta)|^{1/2}$ attains an almost optimal asymptotic performance for both the stochastic, mis-specified and individual setting

The universal probability in this case is horizon independent!

Using Probability Calculus

- If a prior $w(\theta)$ on the model class is postulated

A universal probability for prediction:

$$\int_{\theta} \frac{w(\theta)P_{\theta}(x^{t-1})d\theta}{\int_{\theta'} w(\theta')P_{\theta'}(x^{t-1})d\theta'} P_{\theta}(x_t = x|x^{t-1}) = \int_{\theta} w(\theta|x^{t-1})\, P_{\theta}(x_t = x|x^{t-1})d\theta$$

Essentially, in many cases the prior is not dominant.

Each hypotheses is weighted according to its fitness to the observed past data

Related to **exponential weighting** in learning theory

Analogous result in "universal portfolios"

More on the Bayesian Solution

- Consider two completely different probability measures: $P_1(\underline{x}), P_2(\underline{x})$

  Suppose $\quad D(P_1||P_2) \gg 1, \quad D(P_2||P_1) \gg 1$

- Consider the Bayesian mixture $Q(\underline{x}) = \frac{1}{2}\left(P_1(\underline{x}) + P_2(\underline{x})\right)$

More on the Bayesian Solution

- Consider two completely different probability measures: $P_1(\underline{x}), P_2(\underline{x})$

  Suppose $\quad D(P_1||P_2) \gg 1, \quad D(P_2||P_1) \gg 1$

- Consider the Bayesian mixture $Q(\underline{x}) = \frac{1}{2}\left(P_1(\underline{x}) + P_2(\underline{x})\right)$

  Interestingly: $\quad D(Q||P_1) \leq 1, \; D(Q||P_2) \leq 1$

  Furthermore: $\quad -\log Q(\underline{x}) \leq \min_i [-\log P_i(\underline{x})] + 1 \quad \forall \underline{x}$

Example

- Binary data, Bernoulli Hypothesis Class:

$$P_\theta\,(x^n) = \ \theta^{n_0}\,(1-\theta)^{n_1}$$

Asymptotically, capacity achieving prior: Dirichlet$({}^{1}/_{2})$ – Jeffreys' prior

Thus, after observing (n_0, n_1): $P_U\left(x_{n+1} = 0 | x^n\right) = \frac{n_0+0.5}{n+1}$

In general – intuition – sample the parameter space at $\sim {}^{1}/_{\sqrt{n}}$ resolution

Learning Universally

`Batch' Supervised Learning

- $\boldsymbol{x}^{N-1}, \boldsymbol{y}^{N-1}$ are the training set

- Predict a new outcome y_N, given a new data sample x_N.

Prediction: $b_N = P(\cdot\,|x_N; x^{N-1}, y^{N-1})$

- The loss $\ell(b_N, y_N) = -\log P(y_N|x_N; x^{N-1}, y^{N-1})$ is the "test error".

The (expected) regret is the "generalization error"

\`Batch' Supervised Learning

- $\boldsymbol{x}^{N-1}, \boldsymbol{y}^{N-1}$ are the training set

- Predict a new outcome y_N, given a new data sample x_N.

  Prediction: $b_N = P(\cdot\,|x_N; x^{N-1}, y^{N-1})$

- The loss $\ell(b_N, y_N) = -\log P(y_N|x_N; x^{N-1}, y^{N-1})$ is the "test error".

  The (expected) regret is the "generalization error"

Stochastic setting: Data and outcome are generated obeying some $P_\theta(y^N|x^N), \theta \in \Theta$. Several assumption on x^N.

Consider the "expected **test regret**":

$$\boxed{\min_q \max_\theta \mathbb{E}_{P_\theta} \log \frac{P_\theta(y_N|x_N\,)}{q(y_N|x_N; x^{N-1}, y^{N-1})}}$$

`Batch' Supervised Learning

- $\boldsymbol{x}^{N-1}, \boldsymbol{y}^{N-1}$ are the training set
- Predict a new outcome y_N, given a new data sample x_N.

Prediction: $b_N = P(\cdot\,|x_N; x^{N-1}, y^{N-1})$

- The loss $\ell(b_N, y_N) = -\log P(y_N|x_N; x^{N-1}, y^{N-1})$ is the "test error".

The (expected) regret is the "generalization error"

'PAC' setting: x^N, y^N are i.i.d., according to some unknown distribution $P(x^n, y^n)$, where $P(y^n|x^n)$ is not necessarily in the family.
PAC learning consider the probability over the training that "**expected test regret**" is greater than ε. **Standard PAC results do not work for log-loss!**

Consider instead the "mis-specified expected **test regret**":

$$\boxed{\min_q \max_P \mathbb{E}_P \log \frac{P_{\theta(P)}(y_N|x_N)}{q(y_N|x_N; x^{N-1}, y^{N-1})}}$$

`Batch' Supervised Learning

- $\boldsymbol{x}^{N-1}, \boldsymbol{y}^{N-1}$ are the training set
- Predict a new outcome y_N, given a new data sample x_N.

Prediction: $b_N = P(\cdot\,|x_N; x^{N-1}, y^{N-1})$

- The loss $\ell(b_N, y_N) = -\log P(y_N|x_N; x^{N-1}, y^{N-1})$ is the "test error".

The (expected) regret is the "generalization error"

Individual setting: x^N, y^N are individual sequences.

Learn a universal assignment that outperforms any model $P_\theta(y^N|x^N), \theta \in \Theta$. **Does it makes sense?**

Let

$$\hat{\theta}(z^N, x, y) = \arg\max_\theta p_\theta(y^N, y|x^N, x) = \arg\max_\theta \left[p_\theta(y|x) \prod_{t=1}^{N} p_\theta(y_t|x_t) \right]$$

min over q, max over z^{N+1}=(x^{N+1}, y^{N+1}) of:

$$R_{perm}(q, z^{N+1}) = \frac{1}{(N+1)!} \sum_{\tilde{z}^{N+1}=perm(z^{N+1})} \log \left(\frac{p_{\hat{\theta}(z^{N+1})}(\tilde{y}_{N+1}|\tilde{x}_{N+1})}{q(\tilde{y}_{N+1}|\tilde{x}_{N+1}; \tilde{z}^N)} \right) = R_{LOO}(q, z^{N+1}) = \frac{1}{N+1} \sum_{t=1}^{N+1} \log \left(\frac{p_{\hat{\theta}(z^{N+1})}(y_t|x_t)}{q(y_t|x_t; z^{(N+1)\setminus t})} \right)$$

Some Results

(Fogel-F '18 and after, Vituri-F '24)

- **Stochastic case:**

  A Bayesian mixture over Θ with a prior $w(\theta)$

$$\max_{w} I(\Theta; Y_N | Y^{N-1}, X^N) \triangleq pC_N(\Theta)$$

Asymptotically not Jeffreys'; behaves as k/2N for "nice" k-parameters class

- **Mis-specified case:**

  A mixture over the large class Φ with prior $w(\phi)$ that concentrates "near" Θ

$$\max_{w} \left[I(\Phi; Y_N | Y^{N-1}, X^N) - \sum w(\phi)\, D\left(P_{\phi}(y_N|x_N) || P_{\phi(\theta)}(y_N|x_N)\right)\right] \triangleq pF_N(\Phi, \Theta)$$

- **Individual case:**

  Approximated by the pNML (also cNML): $\boldsymbol{q(y|x, x^N, y^N)} = \dfrac{P_{\theta(z^N,x,y)}(y|x; x^N, y^N)}{\sum_{y'} P_{\theta(z^N,x,y')}(y'|x; x^N, y^N)}$

  With regret: $p\Gamma_N(\Theta) \triangleq \log \sum_y \boldsymbol{P_{\theta(z^N,x,y)}(y|x;\, x^N, y^N)}$

$$pC_N(\Theta) \leq pF_N(\Phi, \Theta) \leq p\Gamma_N(\Theta)$$

In recent work - $\boldsymbol{R_{perm}}$ (for $p\Gamma_N(\Theta)$) was shown to behave as k/N for "nice" k-parameters class

Role of Training
(Fogel-F 2018, Painsky-F 2021, Rosas et. Al.)

1. The role of training is to focus on a smaller, restricted class $\Theta_r \subset \Theta$

   Then apply the standard prediction algorithms (in all settings) on Θ_r

For example, pNML is a special case, where the restricted class is defined by the high likelihood of the training data

2. Efficient compression of the training data: overhead f(n), f'(n) -> 0

   implies successful prediction of the test sample

An alternative Information Theoretic Framework to Machine Learning

- Alternative to "PAC" Learning:

  Use Capacity instead of VC-Dimension or Radamacher Complexity

It turns out that $I(Y^N; \theta | x^N)$ may be rather easy to bound:

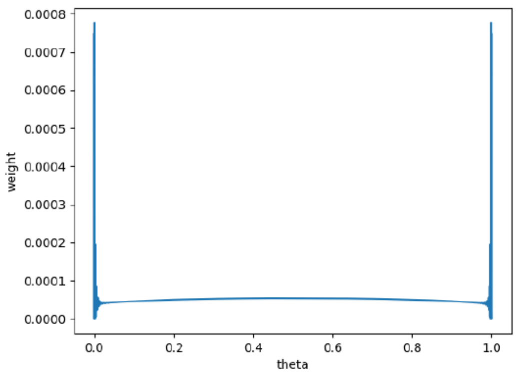
- Consider hypotheses classes that first assign x into one of two groups, and then use a different probability $p_j(y)$ for each group.
- Assume that the VC dimension of the partitions into groups is some finite d. Using Sauer's lemma, we can bound the number of possible partitions by $(\frac{eN}{d})^d$.
- The number of bits needed to represent the best partitions is thus bounded by $d \log(eN)$.
- The number of bits needed to represent the best probability assignments, assuming binary y, is just bounded by $2 \log(N)$.
- Thus, we get $R \leq \frac{d \log(eN) + 2 \log(N)}{N}$.

Interesting simple example

- Again, binary data, Bernoulli Hypothesis Class:

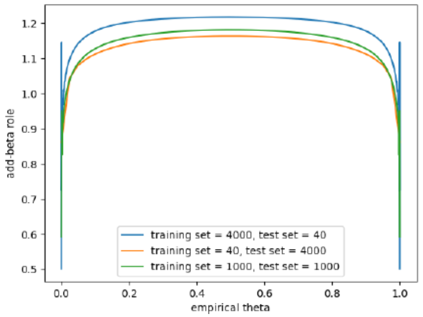
$$P_\theta\,(x^n) = \theta^{n_0}\,(1-\theta)^{n_1}$$

Conditional capacity achieving prior can be found numerically:



deviate considerably from Jefferys prior, $w(\theta) \sim \frac{1}{\sqrt{\theta\cdot(1-\theta)}}$,

Thus, after observing (n_0, n_1): different from "add-β". Equivalent β is



If the edges (0,1) are neglected, get "add-β" with $\beta = 1 + \sqrt{1/6}$

Large Class of Models

Universality w.r.t a large class of models

- A large class Θ of models may correspond to "unbounded complexity"

- Suppose it can be arranged as a union of simpler classes: $\bigcup_i \Theta_i$ each with a universal probability Q_i and complexity C_i, F_i or Γ_i

- A typical situation is hierarchy of models $\Theta_1 \subseteq \Theta_2 \subseteq \cdots$
  - Markov sources of growing order
  - Finite-state models with a growing number of states. In this case the class Θ_s of model with s states is itself a union: of model classes $\Theta_s = \bigcup_i \Theta_{s,i}$

Finite State and Markov Models

- Finite state with $|S|$ states:

  The model class is defined by s_0 and the transition function f:

$$s_{t+1} = f(s_t, x_t)$$

The model class has $|S||X|$ parameters defining $P(x|s)$.

- Markov model of order k: $s_t = \ x_{t-1}, \dots, x_{t-k}$. Has $|\mathcal{X}|^k$ parameters

FS Complexity of an Individual Sequence

- Lempel-Ziv 78 defined a Finite-State complexity measure of an individual sequence.
- The measure is the codelength (or log-loss) attained by any Finite-State machine of any size for that (infinite) sequence.
  - first, find the best FS model with S states for x^n (including its parameters)
  - then, consider the normalized loss and let $n \to \infty$
  - finally, let $S \to \infty$

$$\mathcal{U}(x) = \lim_{S\to\infty} \limsup_{n\to\infty} \frac{1}{n} \min_{f \in \mathcal{F}_S} \mathcal{L}_f(x^n)$$

set of S-state machines

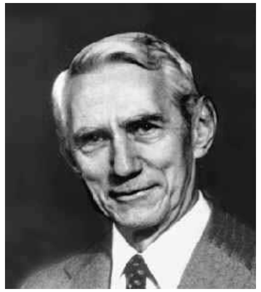
Best model with FSM f

FS predictability of the (infinite) sequence x

Complexity is scaled with the amount of data

- This quantity is efficiently achievable by the Lempel-Ziv data compression scheme

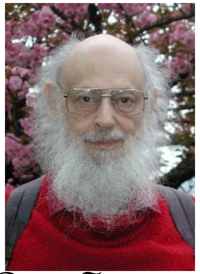
- While not as general as Kolmogorov's complexity, it is computable!



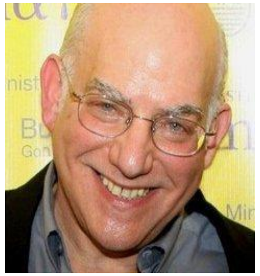
Claude Elwood Shannon

Entropy: Complexity measure in the stochastic case

Andrei Nikolaevich Kolmogorov

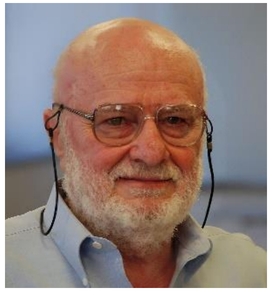


Ray Solomonoff



Gregory Chaitin

Algorithmic complexity of individual sequence



Abraham Lempel

Jacob Ziv

Finite-State complexity of individual sequence

Attained by Lempel-Ziv 78 scheme

Lempel-Ziv (LZ) universality

- LZ attains the entropy, if the source is ergodic (stochastic setting), or attains the FS complexity (individual setting):

  …. **by essentially increasing the "model size" as more data is available***

- How fast?
  - **Slow** $O\left({}^{1}\!/_{\log n}\right)$ convergence
  - The reason: **the "dictionary" keeps increasing even if the data exhibits a simple model**
  - Practical methods of adaptive dictionary construction improves the rate!

- **Need a method that "learn" the complexity!**

* Some may use this approach to explain learning – "scaling laws"

Twice/Multiple/Hierarchical Universality

Twice Universality

- Model classes with different complexity (different C_i, F_i or Γ_i)
- The source/explanation comes from a model in one of the classes

Twice Universality

- Model classes with different complexity (different C_i, F_i or Γ_i)

- The source/explanation comes from a model in one of the classes

  - Again, may consider $\Theta = \bigcup \Theta_i$; **however, this will lead to a large "redundancy"**

  - Depending on the data size, take into account Θ_i's that have negligible redundancy. Consider more classes as more data is available – like in LZ

Twice Universality

- Model classes with different complexity (different C_i, F_i or Γ_i)
- The source/explanation comes from a model in one of the classes
  - Again, may consider $\Theta = \bigcup \Theta_i$; **however, this will lead to a large "redundancy"**
  - Depending on the data size, take into account Θ_i's that have negligible redundancy. Consider more classes as more data is available – like in LZ

Proposed Preferred option:
Twice Universality (originally, Ryabko, 1985)
Generalized to **Multiple Universality**

Twice Universality

Universal with respect to the choice of the model class!

- Suppose Q_i is the universal probability of the class Θ_i

Twice Universality

Universal with respect to the choice of the model class!

- Suppose Q_i is the universal probability of the class Θ_i

Find a "twice universal" $\boldsymbol{Q}$ that can represent all the $\boldsymbol{Q_i}$'s

Seems like a good approach

For example, find $C_{TU} = \min_{Q} \max_{i} D(Q_i || Q)$

However, might get $\boldsymbol{C_{TU} \gg C_i}$ ***Bad…***

Twice Universality

Possible Solution for the last: Multiplicative regret. Consider: $\frac{\log Q}{\log Q_i}$

- In the individual case: Let $i(x^n) = \arg\max_i Q_i(x^n)$

Solve: $\boxed{\min_Q \max_{x^n} \frac{\log Q(x^n)}{\log Q_{i(x^n)}(x^n)}}$

Twice Universality

Possible Solution for the last: Multiplicative regret. Consider: $\frac{\log Q}{\log Q_i}$

- In the individual case: Let $i(x^n) = \arg\max_i Q_i(x^n)$

Solve: $\boxed{\min_Q \max_{x^n} \frac{\log Q(x^n)}{\log Q_{i(x^n)}(x^n)}}$

- Solution: $Q_{TW}(x^n) = Q^{\alpha}_{i(x^n)}(x^n) = Q^{\beta}_{i(x^n)}(x^n) Q_{i(x^n)}(x^n)$
  where $\alpha \geq 1$ is chosen so that $\sum_{x^n} Q_{TW}(x^n) = 1$

Note that $-\log Q_{TW}(x^n) = -\log Q_i(x^n) + [-\beta \log Q_i(x^n)]$

The extra loss is proportional to the "universal codelength" of Θ_i

Yet, a better solution: Multiple/Hierarchical Universality

- Large $\boldsymbol{C_{TW}}$ since there are “too many classes”
  - Add another universality layer:
    Three-times universality, multiple universality

- Eventually, in the last universality layer, can make the extra cost “proportional” to the accumulated complexity

Multiple Universality: Canonical Example

The universal representation of the integers (or the “complexity”):

We wish to represent an integer universally, with number of bits proportional to its binary representation**. <u>Range can be all the integers!</u>**

Multiple Universality: Canonical Example

The universal representation of the integers (or the "complexity"):

We wish to represent an integer universally, with number of bits proportional to its binary representation. **Range can be all the integers!**

- Consider the classes [2,3], [4,5,6,7],…. Class Θ_i contains 2^i models.
- Simplest case – each model is deterministic on some value. In this case, the universal probability of each class :

$$Q_i = 2^{-i} \Rightarrow -\log Q_i = i = \lfloor \log n \rfloor, \ n \text{ the corresponding integer}$$

Multiple Universality: Canonical Example

The universal representation of the integers (or the "complexity"):

We wish to represent an integer universally, with number of bits proportional to its binary representation. **Range can be all the integers!**

- Consider the classes [2,3], [4,5,6,7],…. Class Θ_i contains 2^i models.
- Simplest case – each model is deterministic on some value. In this case, the universal probability of each class :

$$Q_i = 2^{-i} \Rightarrow -\log Q_i = i = \lfloor \log n \rfloor, \ n \text{ the corresponding integer}$$

- It turns out that a multiplicative universal probability should use $\alpha = 2$

Thus, $Q_{TW} = 2^{-2i} \Rightarrow$ universal code for the integer requires $2\lfloor \log n \rfloor$ bits

- *With an extra bit to specify the integer 1, get Elias universal representation of the integers*

Can repeat the process for "multiple universality" (or "hierarchical universality"). Attain a representation of the integer $\boldsymbol{k}$ with $log^* \, k$ **bits**

Elias' Codes

Fig. 1. Partition into groups - gamma code

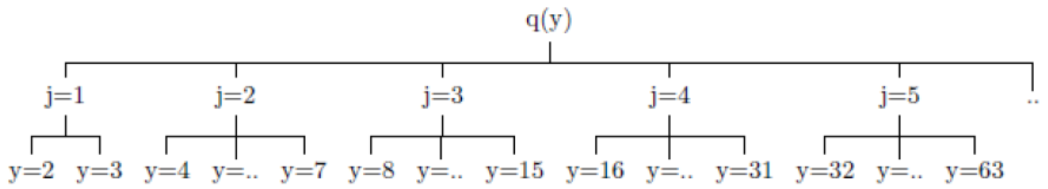


Fig. 2. Partition into groups - delta code

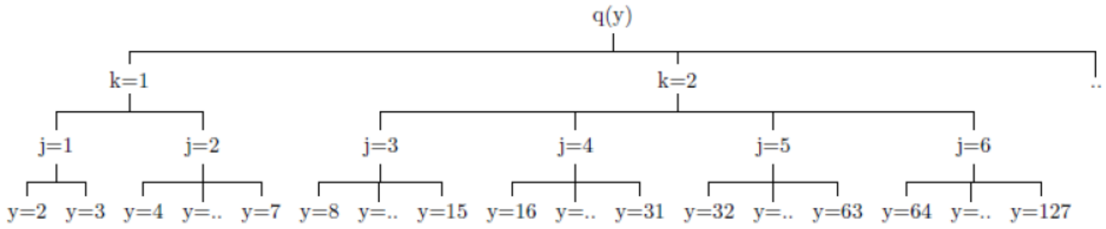
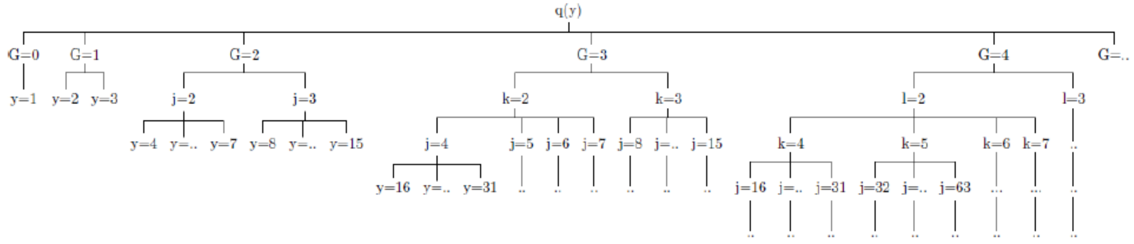


Fig. 3. Partition into groups - omega code



Markov Models

- The basis of the Minimum Description Length (MDL) principle

A universal code of a binary sequence x^n for the class of k^{th}- order Markov model requires:

$$L_k(x^n) = \widehat{H_k}(x^n) + \frac{2^k}{2n}\log n \text{ bits/symbol}$$

where $\widehat{H_k}(x^n)$ is the k^{th}- order Markovian empirical entropy – ML solution in the class

MDL principle: choose the model that minimizes $L_k(x^n)$

- A "twice universal" two-part code (actually 3-part code..)

$$L_{TU}(x^n) = \min_k [L_k(x^n) + \log^* k]$$

- The extra "cost" for not knowing the class is negligible w.r.t the cost of not knowing the model in the class

Non-uniform convergence to the minimal empirical entropy

Twice/Multiple Universality

- More generally - a Bayesian solution with a prior that is inversely proportional to the class complexity

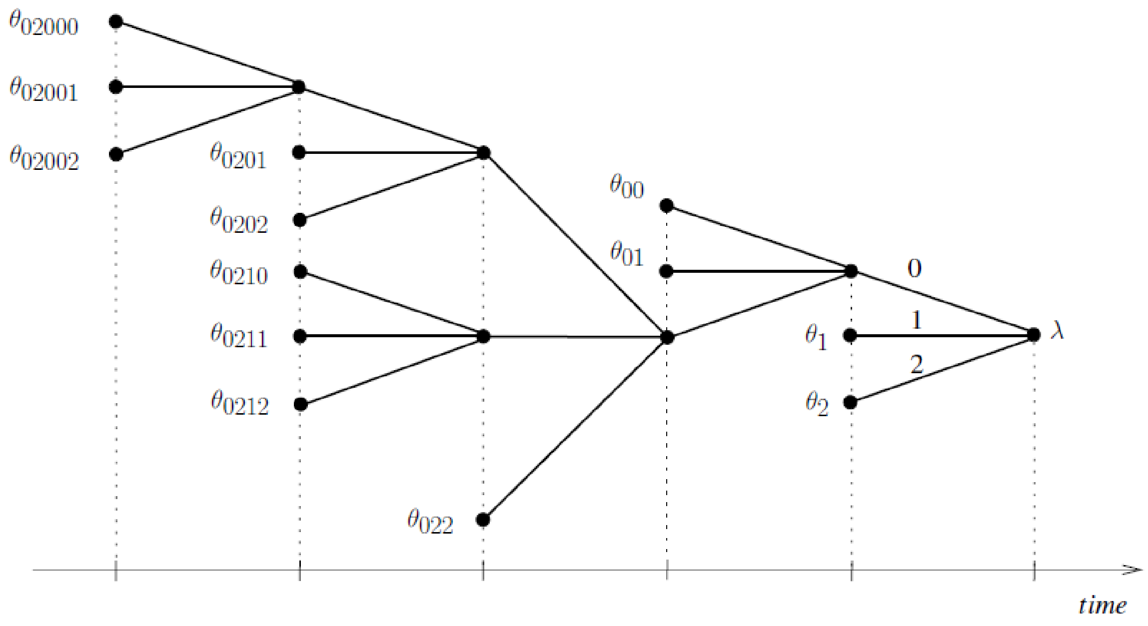
Practical success of Twice Universal Coding:

- Context-tree weighting (CTW, Willems et. al, 1996)
- Prediction by partial matching (many authors)
- Plug-in: A universal finite memory source (Weinberger, Rissanen, F – 1995)

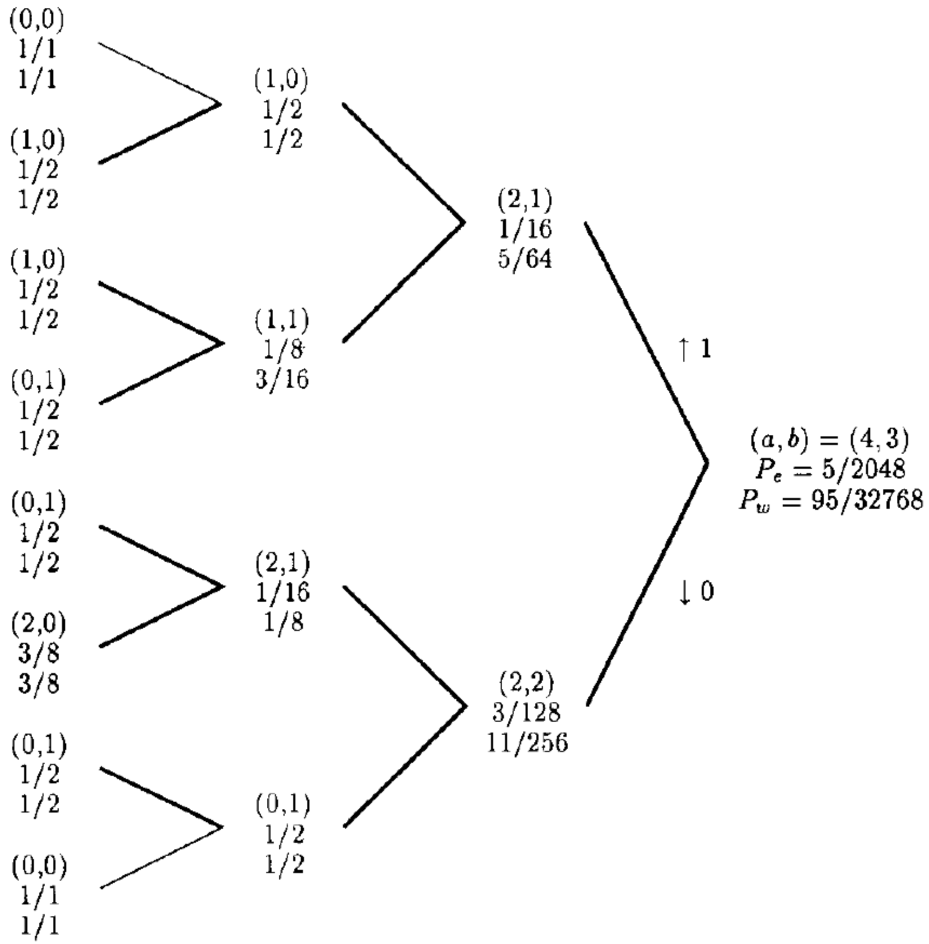
Underlying model classes: variable order Markov models

Variable Order Markov Models
(unifilar sources)

- Unifilar models:



More on CTW



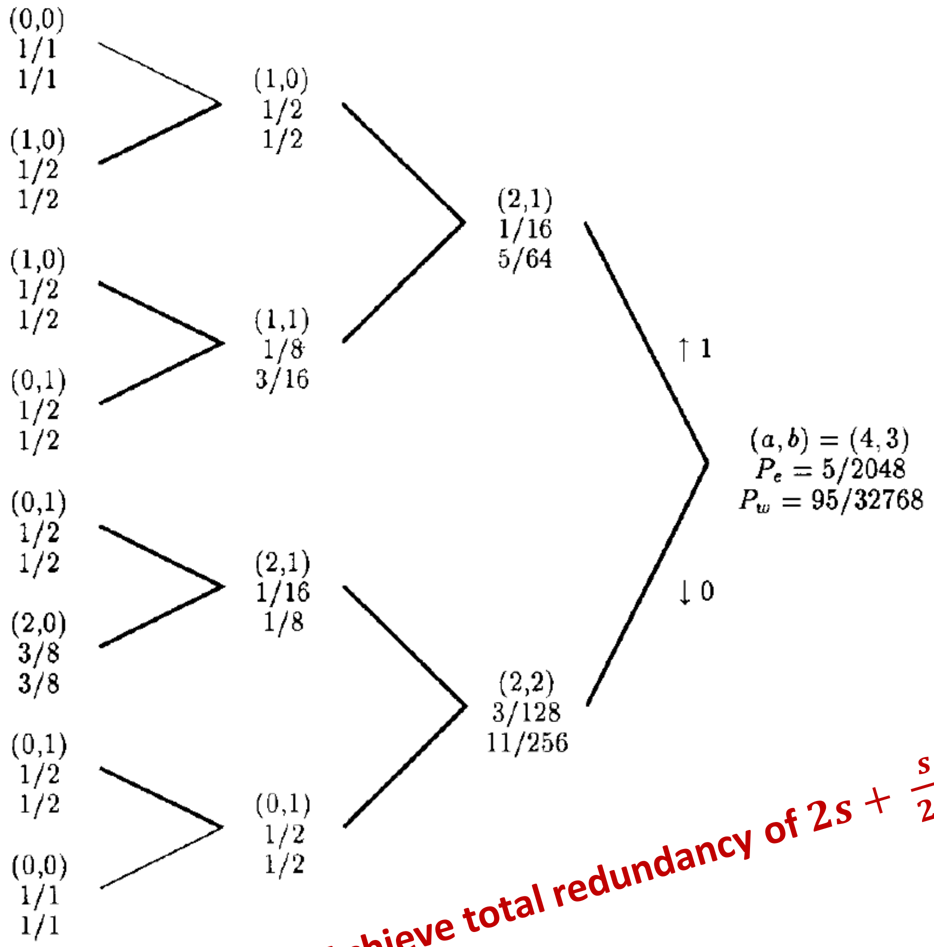
Weight all possible sub-trees of context to generate the multiple universal probability

Effective weight of a tree with S leaves: $2^{-(2S-1)}$
Resulting extra redundancy $2S - 1$

Comparable (and smaller) than the redundancy of Q_{tree} of ${}^{S}/_{2} \log n_S$

Recently extended, analyzed and made more efficient computationally – Kontoyiannis 2020

More on CTW



Achieve total redundancy of $2s + \frac{s}{2n} \log \frac{n}{s} + 1$ bits/symbol

Weight all possible sub-trees of context to generate the multiple universal probability

Effective weight of a tree with S leaves: $2^{-(2S-1)}$
Resulting extra redundancy $2S - 1$

Comparable (and smaller) than the redundancy of Q_{tree} of $\,^{S}/_{2} \log n_S$

Recently extended, analyzed and made more efficient computationally – Kontoyiannis 2020

Multiple Universality Interpretation

Consider a 3-part coding scheme:

- Code the number of leaves $|s|$ using Elias recursive codes.
- Given $|s|$, code the exact tree - there are $C(s) \sim \frac{4^s}{s^{3/2}\pi}$ trees with s leaves.
- Code the probabilities at each leaf using KT-estimator.

Works for any memory size!

The redundancy is slightly better than the bound for CTW, R_{CTW}:

$$R_{3pc}(y^n) = 2|s| - 1.5\log(|s|) + \log^*(s) + \frac{|s|}{2}\log(\frac{n}{|s|}) + O(1)$$

This analysis also gives insight to the $2|s|$ term in R_{CTW}.

Large Alphabet

- Large alphabet $d \gg n$. d may even be unknown or infinite..

- Hierarchy according to the alphabet size:

$$\log^* k + \log\binom{d}{k} + \frac{k}{2}\log n + \left(-\log p_{\theta_1,\theta_2,\ldots,\theta_k}(y^n)\right)$$

works for $k \ll n$

In the predictive distribution d disappear; get Ristad's law of succession!

One consequence – predictive probability of "unseen" symbol:

$$\frac{k(k+1)}{n^2+n+2k}$$

Large Alphabet (2)

- Large alphabet $d \gg n$. d may even be unknown or infinite..
- Hierarchy according to the empirical counts:

$$\log Part(n) + \log \binom{d}{d_1 \ldots d_n} \; + (-\log p_{emp}(y^n))$$

In the predictive distribution d disappear; get close to Good-Turing law of succession!

A consequence – predictive probability of "unseen" symbol:

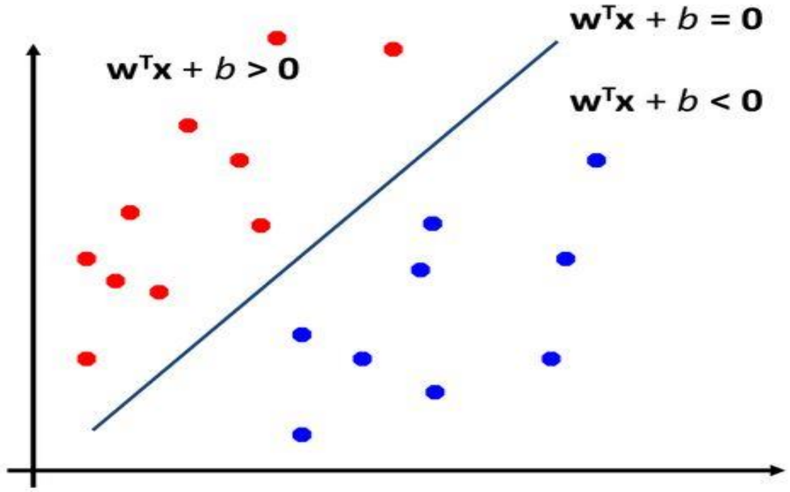
$$\sim \frac{d_1}{n(d-k)}$$

**This hierarchy weights more low entropy empirical distribution,
while the previous hierarchy weights more small alphabet size**

Perceptrons (Linear Separators)

Linear Separators

- Binary classification can be viewed as the task of separating classes in feature space:



$f(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x} + b)$

Lower dimensional separators can replace higher dimensional with large margin

Multiple universality over the dimension

Perceptrons and other Linear Models:
An alternative Hierarchy

Consider the general "model class":

$$\{p_\theta(y|\underline{x}) = f(y; \underline{\theta}^T \underline{x}\}$$

where $f(\cdot)$ is a general stochastic function and $\underline{\theta}, \underline{x}$ are d-dimensional vectors, $d \gg n$ can be very large

This case include perceptrons, linear regression, logistic regression and many more model. OVERPARAMETERIZED!

Perceptrons and other Linear Models:
An alternative Hierarchy

The embedded model classes: All subsets of $\{\theta_1, \theta_2, \ldots., \theta_d\}$

Arrange the classes according to their cardinality:

d classes with a single parameter, $\binom{d}{2}$ with 2 parameters, and so on..

A multiple-part description for this hierarchy:

Specify K the number of parameters, then which class of K parameters, then the K parameters and finally the data given this description. This requires:

$$\log K^* + \log\binom{d}{K} + \frac{K}{2}\log n + \left(-\log p_{\theta_{i_1},\theta_{i_2},\ldots,\theta_{i_K}}(y^n \,|\underline{x}^n)\right)$$

A slightly shorter "codelength" will be obtained by the appropriate mixtures

Linear Regression Example: Polynomial fit

Linear regression solution is

$$\hat{\theta} = \left(X_N^\top X_N\right)^{-1} X_N^\top Y_N = \sum_{i=1}^{M} \frac{u_i^\top}{h_i^2} X_N^\top Y_N \tag{1}$$

The pNML solution is

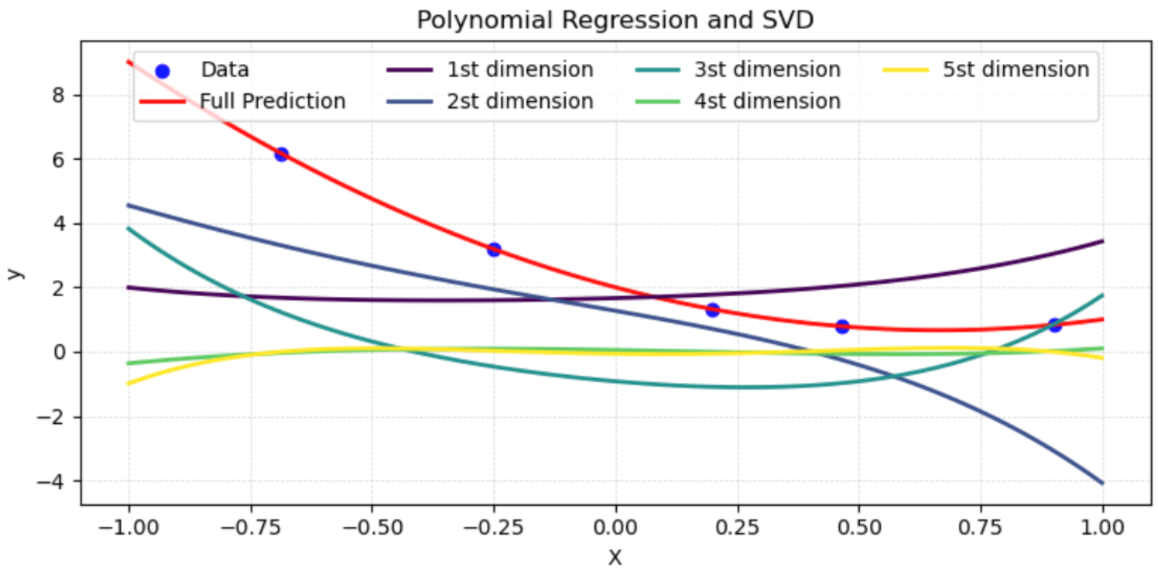
$$p(y|x,\theta) = \frac{1}{\sqrt{2\sigma^2 K^2}} exp\left\{-\frac{(y - x^\top \hat{\theta})^2}{2\sigma^2 K^2}\right\} \tag{2}$$

We define the summation using n dimensions $i_1, i_2, ..., i_n$ as

$$\theta_{i_1,i_2,...,i_n} = \sum_{i \in (i_1,i_2,...i_n)} \frac{u_i^\top}{h_i^2} X_N^\top Y_N \tag{3}$$

To combine the predictions of the multiple model class

$$\begin{aligned} p(y|x) = & \sum_{i_1=1}^{M} p(y|x,\theta_{i_1}) \\ & + \sum_{i_1=1}^{M} \sum_{i_2=i_1}^{M} p(y|x,\theta_{i_1,i_2}) \\ & + \sum_{i_1=1}^{M} \sum_{i_2=i_1}^{M} \sum_{i_3=i_2}^{M} p(y|x,\theta_{i_1,i_2,i_3}) \\ & + ... \\ & + \sum_{i_1=1}^{M} \sum_{i_2=i_1}^{M} ... \sum_{i_M=i_{M-1}}^{M} p(y|x,\theta_{i_1,i_2,...,i_M}) \end{aligned} \tag{4}$$



Multiple Universality Linear Regression

With Random Feature vectors:

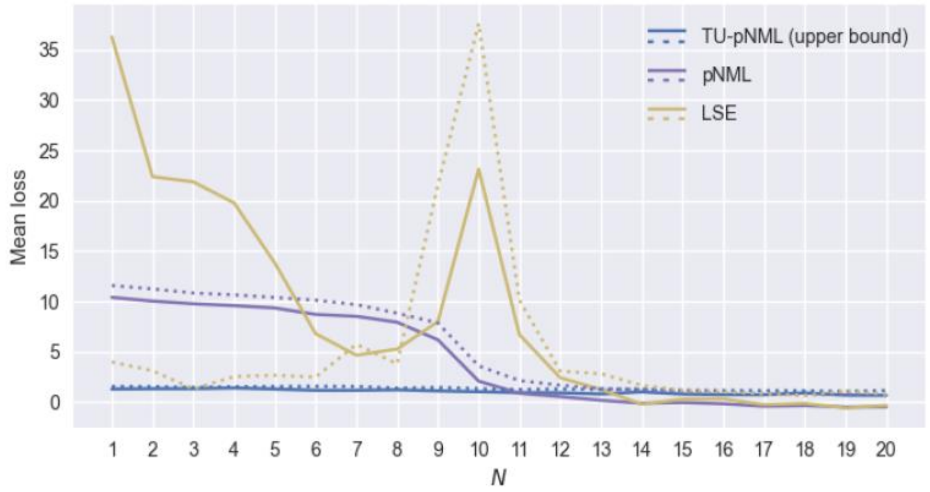


Fig. 3. Mean loss for $d = 10$ as a function of $n = 1, \ldots, 20$ over 50 iterations, using $\sigma^2 = 0.01$ (solid) and $\sigma^2 = 0.1$ (dashed).

Need computationally efficient (recursive?) algorithm!

Are Deep Neural Networks Hierarchically Universal?

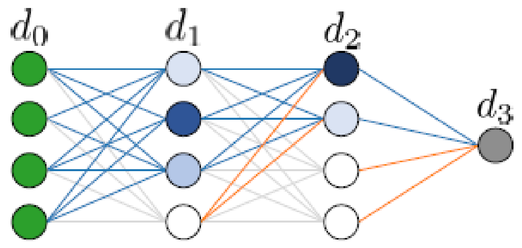
Need to find the right hierarchy structure..

Sparsity of layer inputs, after the ReLU non-linearity

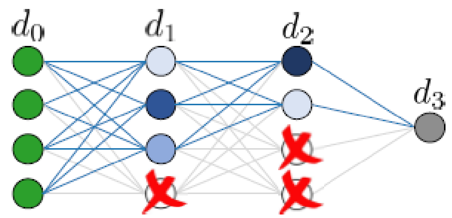
Manifolds of various dimensions

.. and so on

A large Network as a Union of Small Networks



A Large Network



An embedded smaller Network

- Specify the number of nodes in each layer;
- Then, specify a small networks with these number of nodes;
- Then specify the parameters associated with this smaller network;
- Finally encode the data given this description. This requires:

$$\sum \log d_i + \Sigma \log \binom{d_i}{d_i^*} + \Sigma \frac{d_i^* d_{i-1}^*}{2} \log n + (-\log p_{small\ network} \quad (y^n \,|\underline{x}^n))$$

With fully connected network do not care about the node choice. Do not need 2nd term

Experiment with Hierarchy of Small Networks

- Given a large networks. Large hypothesis class
- Randomly select sub-networks of various “capacities”. Total number of parameters are equivalent to the large network size
- Approximate the universal predictor of each sub-network by the ERM with SGD
- Perform “multiple universality” over these “universal predictors”:
  - Average the predictors – ensemble averaging, however:

  Weights reflect both –
  - The complexity of the sub-network: larger networks are penalized
  - An exponential weighting according to the fitness to the training

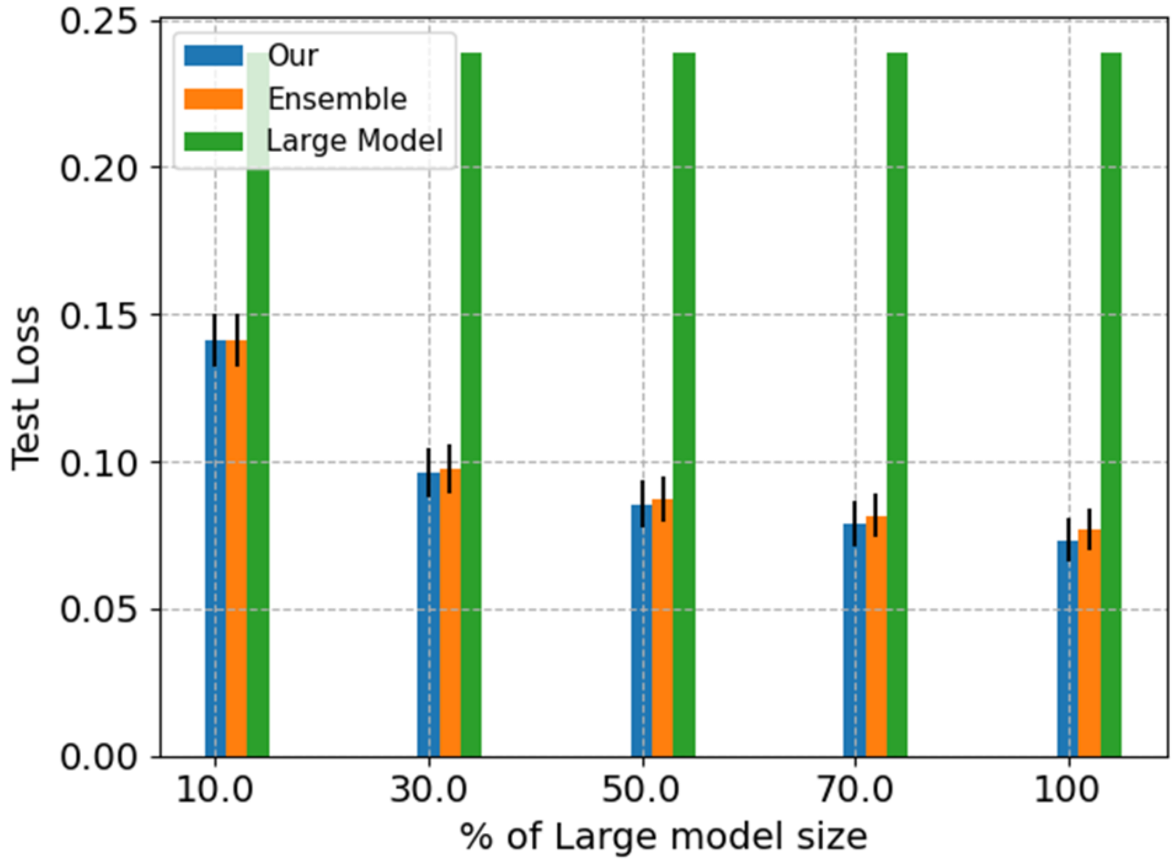
Results

of parameters Large model 266610
Large model - Train Loss=0.0033 Test Loss=0.2460
##############################
Prob = 1, **pool of models = 595**
##############################
--**SMALLEST MODELS** ---

Num of params = [4015, 4095, 4175, 4255, 4335, 4415, 4495, 4575, 4655, 4735, 4815, 4895, 4975, 5055, 5135, 5215, 5295, 5375, 5455, 7965, 8070, 8175, 8280, 8385, 8490, 8595, 8700, 8805, 8910, 9015, 9120, 9225, 9330, 9435, 9540, 9645, 9750, 9855]
Train Loss = [0.3954, 0.2753, 0.24759, 0.51102, 0.8354, 0.25432, 0.23593, 0.36226, 0.24435, 0.23079, 0.235, 0.2334, 0.23162, 0.23547, 0.23665, 0.22929, 0.22265, 0.24359, 0.23716, 0.19122, 0.12353, 0.16399, 0.12197, 0.11127, 0.08941, 0.09461, 0.15591, 0.15254, 0.09045, 0.09315, 0.08318, 0.09299, 0.08623, 0.10994, 0.0675, 0.06373, 0.06295, 0.06052]
Test Loss = [0.43543, 0.31145, 0.2936, 0.56065, 0.90121, 0.31526, 0.29546, 0.40616, 0.29731, 0.29901, 0.30029, 0.29228, 0.29415, 0.3073, 0.31207, 0.2949, 0.3018, 0.33312, 0.31027, 0.25704, 0.20413, 0.23476, 0.2006, 0.20284, 0.17873, 0.18968, 0.23554, 0.23159, 0.20639, 0.20361, 0.20059, 0.19853, 0.18341, 0.20173, 0.19246, 0.21829, 0.20162, 0.20023]
Eval Loss = [0.3819, 0.2537, 0.2416, 0.4781, 0.804, 0.2628, 0.2276, 0.351, 0.2427, 0.2365, 0.2361, 0.2325, 0.2461, 0.2311, 0.2418, 0.2333, 0.2282, 0.2601, 0.248, 0.1948, 0.1741, 0.1959, 0.1756, 0.1661, 0.1574, 0.1695, 0.1877, 0.1854, 0.1622, 0.1577, 0.1662, 0.187, 0.1626, 0.172, 0.1525, 0.1459, 0.1481, 0.1565]

Testing AVERAGED Ensemble of 38 models:
Test: Loss: 0.1425, Accuracy: 7167/7500 (95.56%)

Testing WEIGHTED AVERAGED Ensemble of 38 models:
Test: Loss: 0.1409, Accuracy: 7189/7500 (95.85%)



Yet, there is still an
Elephant in the room:

What is the “Hierarchy”?
Is there a “right” hierarchy?

This might be undecidable

It definitely may require
enormous computation

THANKS!